# METHOD AND APPARATUS FOR AUTHENTICATING AND VERIFYING COMMUNICATION ON A NETWORK OF GAMING DEVICES

## CROSS REFERENCE TO RELATED APPLICATIONS

This application claims priority from US Patent Application No. 10/256,949, "SYSTEM FOR AWARDING A BONUS TO A GAMING DEVICE ON A WIDE AREA NETWORK," filed September 27, 2001.

## 1. Field of the Invention

The present invention relates to awarding a bonus on a network of gaming devices, and—more particularly—awarding such a bonus on a wide area network.

## 2. Background of the Invention

Many casino operating companies own multiple casinos in several locations. It has proved to be advantageous for such companies to devise player loyalty promotions that span these properties. Historically, multi-property player management systems accumulate player activity across all properties. For example, many casinos track a player's total wagers, theoretical win, actual win, complementary balance, player point balance, and other key player behavior statistics across all casinos managed by that operator.

This information may be used to determine a global worth of the player to the operator, as opposed to a single playing location. Many casino operators accomplish this by dividing their player database into tiers, with the highest tier being the most valuable players and the lowest tier being the least valuable players from the casino's perspective. The casino typically uses a statistic such as theoretical win to establish player worth. Once this global player worth is established, casinos target promotional offers commensurate a player's worth. The promotions are designed to entice further play at one or more of the operator's casinos.

The promotions usually take advantage of the multi-property nature of the business, such as allowing redemption of promotions at all casinos; targeting redemptions to a single property that might have more capacity than other properties; and using a complementary trip to a more desirable property as a promotional offer

All of these generally rely on manual processes for communication of promotional information to the customers and for redeeming promotions. For example, promotional offers are generally communicated via direct mail to qualifying players, or via brochures, signage and literature at a casino. Further, redemption of promotional offers—whether they are cash, complementaries, etc.—is generally done manually at a players-club booth at each casino.

As used herein, the term bonus is an award, e.g., like the promotional offer, given to a player of an electronic gaming machine (EGM). The term bonus herein refers to any such award that is not paid by the device in accordance with its pay table. Such bonuses and systems for implementing them are described in US Patent No. 5,655,961 (the '961 patent) and in US Patent No. 6,319,125, both of which are hereby incorporated herein by reference for all purposes. Also hereby incorporated by reference for all purposes is US Patent No. 6,375,569, which describes a bonus promotion like the one described herein, except implemented at a single casino. A bonus can include an award of cash or machine credits, player points, or complementary amenities.

In a first implementation of the present invention, as participating EGMs are played, a user-selected percentage of the play is added to a common bonus pool. When the pool reaches a randomly selected level of play, between specified minimum and maximum numbers, a winner is randomly selected. The award can be a fixed cash amount, a cash amount linked to the bonus pool total or a non-cash prize. Prizes that are typically smaller than the winning bonus can also be awarded to non-winning players.

In the preferred embodiment, a master server, located at one of a plurality of participating casinos, communicates over a wide area network with slave servers located at each participating casino. The preferred embodiment may support a master server, up to 32 slave servers, and may accommodate as many as 16 different bonus pools that operate to pay a bonus award to one of the EGMs associated with the respective pools. It should be appreciated that other embodiments could support more or less slave servers and bonus pools. For each bonus pool, the master server selects a winning slave server, which in turn selects a winning EGM.

The preferred embodiment is called the Random Rewards® promotion. The promotion is complete when the amount of play on participating EGMs reaches the randomly selected number between the minimum and maximum numbers, which are specified at the master server. The randomly selected number is called the lucky number. Preferably a player must be issued a player-tracking card to be eligible to participate. Although the invention is not so limited, a typical implementation is for a plurality of casinos that are commonly owned with each recognizing a player-tracking card issued by any of the others. As a result, player activity is tracked—in a known manner—across all of the casinos.

As a security enhancement of the bonus awarding process, as well as other commands in the gaming network, messages detailing the bonusing may be digitally signed, allowing verification of the message prior to trusting the contents.

## Brief Description of the Drawings

Fig. 1 is a schematic diagram of a multi-property bonus system implemented in accordance with the present invention.

Fig. 2 is a flow chart depicting some of the operation of the system depicted in Fig. 1.

Fig. 3 is a schematic diagram of a gaming network with digitally signed commands.

Fig. 4 is a flow chart depicting some of the operation of the elements of a gaming network with digitally signed commands.

## Detailed Description of the Preferred Embodiments

Turning now to Fig. 1, indicated generally at 10 is a system constructed in accordance with the present invention. It includes a wide-area network (WAN) 12 that incorporates a single master server 14. Also included in the WAN is a configuration workstation 15, including a keyboard, monitor, and software, that permits a user of the workstation to configure the master server. Portions of WAN 12 are located at different casinos, one of which is depicted generally at 16. Each casino includes a local area network (LAN), like LAN 18a at casino 16. LAN 18a includes a router 20; a concentrator 24; a slave server 26, which—among other things—tracks carded EGM play in a known manner; a player server 27, which provides messages to displays associated with the EGMs; a key distribution center 29 (KDC), which implements security as will be described; and a plurality of EGMs, only two of which are exemplary slot machines 28, 30. In addition, a bank controller 31 facilitates communication between slot machines 28, 30 and concentrator 24. Bank controller 31 provides the same function for an animation computer 33, which generates animated content that appears on a display 35.

The same components (except for a master server, like master server 14) appear in LANs 18a, 18b, 18c, 18d (not shown) at each of the other casinos (also not shown) on the WAN. It should be appreciated that the master server may also be located at a site remote from any of the participating casinos, or—as in the present embodiment—at one of the casinos; specifically, master server 14 is located at casino 16. Although specified network structure is depicted, the invention can be implemented on any suitable network, regardless of its design or the hardware with which it is implemented.

Router 20 transmits data packets between the master server and each slave server over WAN 12. Depending on topography of the network, a hub could be used in lieu of router 20. Concentrator 24 is a network device similar to a hub that provides communication routing for devices on the network.

As described in the '961 patent, each EGM at each casino includes a communication board. This board, among other things, receives bonus promotion and message information from bonus servers and sends EGM meter information, among other things, to network computers, including the slave server, like slave server 26.

Consideration will now be given to configuration of master server 14, which must be undertaken before play can begin. Master server 14 is configured at workstation 15 with a user-specified contribution rate that EGMs contribute to the growth of a bonus pool, which is called the current pool. In general terms, as participating EGMs are played, the current pool is incremented by the contribution rate multiplied by the total play on the EGMs. When the current pool reaches or exceeds the lucky number, the master server randomly selects a winning slave server. The winning slave server then randomly selects the winning EGM. EGMs that require different denominations to play can be incorporated into the same pool using known techniques to account for the difference in denominations from game-to-game within the pool.

In the multi-casino Random Rewards® promotion, selecting minimum and maximum numbers at master server 14 specifies the range from which the winning number is selected. The range corresponds to a level of play by the participating EGMs; the total current pool results from all plays made on all EGMs participating in the multi-casino promotion. Master server 14 is also configured with a list of all slave servers participating in the promotion. The current pool represents the combined contributions of each slave server. Each slave server pool total further represents the combined contribution of all EGMs participating in the promotion at the associated slave server's casino.

Each slaver server is also configured, primarily by designating which EGMs are linked to a particular bonus pool. The slave servers are also configured using their associated workstation, like server 26 is configured by workstation 15. Each local configuration workstation can be used to configure the master server and the associated local server, but a workstation at one location cannot be used to configure a slave server at another location.

Consideration will now be given to operation of system 10, with reference to the flow chart of Fig. 2, after the master and slave servers are configured as described above. As

EGMs are played in step 32, their respective meters increment in step 34. In a step not depicted in Fig. 2, prior to initiation of a bonus pool, the master server transmits the pool contribution rate—a typical rate being in the .001% to 3% of all coin-in—to each slave server. In step 36, each slave server aggregates meter information from each EGM in its associated casino. Master server 14, whose identity is unknown to the slave servers, sends a display message on a periodic basis—in step 38—that includes the total accrued value of the bonus pool to slave servers. This message operates like a heartbeat. With each such transmission, the slave causes the total bonus pool—accrued from all participating casinos—to be displayed, either on displays associated with each EGM or on overhead display 35, or both. The slave servers respond, in step 40, to each heartbeat message by (a) causing the updated value of the total bonus pool to be displayed and (b) sending their new local pool values to the master server. In the present embodiment, the heartbeat occurs approximately every 5 seconds, but this period can be adjusted upwardly or downwardly.

In the present example, the meter specified at master server 14 to track play is one created for this bonus, namely a carded coin-in meter. This meter counts all coin into the EGM that has a player associated with the EGM as a result of the player's player-tracking card being inserted into a card reader associated with the EGM.

Other meters, such as the coin-in meter, the game meter, the win meter, or another created meter can also be used. Each slave server is updated every half-second with data identifying each EGM whose coin-in meters advanced in the preceding half second and the amount of advance by each. From this the slave server calculates the total coin in for the period and multiplies this times the contribution rate to produce a number equal to the total accrued contribution by that slave server, and therefore its associated casino, since the current pool began accruing. This local pool value is the number transmitted over the WAN in step 40 to the master server.

The slave servers send the total accrued contribution of the casino—i.e., the local pool value—rather than incremental changes to reduce the negative effects caused by communication errors. Sending totals reduces problems arising from packets lost during transmission and obviates the need to reset or synchronize with the master server during increment processing. The master server then compares the new slave pool total with the previous slave pool total (even if power outages or other difficulties prevented transmission of packets for one or more heartbeats) to determine the amount of change, which it then adds to the current pool in step 42. When the current pool reaches the lucky number—in step 44—

the winner is selected. If the lucky number is not reached, play and processing continues as shown.

Once the lucky number is reached, the master server determines a winning slave server. The winning slave server is chosen randomly among slave servers generating local pool increases responsive to the heartbeat that caused the current pool to exceed the lucky number. The random selection, however, is weighted proportionate to the local pool contribution reported by each casino when a winning slave server is chosen. In other words, casinos that generate larger wagers or more play advance the lucky number count more, and are therefore more likely to reach the lucky number.

Once the master server has randomly selected a winning slave server, the master server sends point-to-point messages to each slave server identifying the winning slave server. The winning slave server then identifies a winning EGM also based on a weighted random process. It selects the next half-second of updated play data (described above) to randomly choose a winning EGM from those that had a coin-in event in the selected play data. The random selection is, however, weighted to give a proportionately greater chance of winning to EGMs as a function of the amount of the wager. In other words, the bigger the wager, the more likely it is that an EGM will be selected. The winning EGM pays the bonus, in response to a command from the slave server, and updates its history record. The selected EGM may, but need not, be locked to prevent further play. The EGM may be locked in different ways, depending upon system configuration. First, the user may check a box in a panel appearing on the display of workstation 15 that requires each EGM to lock up whenever the EGM is a Random Rewards® winner. Second, if the box is not checked, an award may or may not be locked up. The EGM manufacturer configures the EGM for a maximum jackpot payable on the EGM credit meter. If the Random Rewards® payment is below that maximum, the payment is paid to the credit meter, from which it can be wagered or cashed out. If the payment is greater than the maximum, the machine is locked up.

In another aspect of the invention, one or more slave servers can be configured to award another bonus, preferably the Celebration Prizes® promotion, when a Random Rewards® winner is selected. In the Celebration Prizes® promotion, the slave server can be designated to award preselected EGMs a bonus when a Random Rewards® winner is selected. Only EGMs that are linked to the bonus pool that includes the winner are eligible for the Celebration Prizes® payment. Each slave server can be set to provide the Celebration Prizes® promotion, or not, and each property that provides it can set criteria establishing which of the

eligible EGMs are awarded a Celebration Prizes® bonus. For example, of the eligible EGMs, Celebration Prizes® winners might be only those with a player's card in, or those whose last bet was a maximum bet, or those who registered a coin drop within a predetermined period of time, or some combination of these or other criteria.

Following payment to the selected EGM, the bonus promotion is continuously repeated. One or more of the bonus pools can be modified by either ending it or making changes to its parameters, such as the contribution rate and the minimum and maximum numbers that define the range from which the lucky number is randomly selected. Commands to modify a pool are entered at workstation 15. The modifications are effective immediately, modifying the current pool. In an alternative embodiment, the modifications are not implemented until after a Random Rewards® winner is selected for the current game associated with the pool to be modified. The commands set a flag, which implements a process after the current game. As a result, the modifications are made commencing with the next game associated with the modified pool.

A new bonus promotion begins by subtracting the lucky number from the current pool, and creating a new current pool by adding a base amount, which is the minimum number for the lower end of the range from which the lucky number is selected; a hidden pool; and a delayed pool (both described below) that are accrued from prior games.

For example, assume that the minimum pool level (base amount) is $1000 and the hidden pool increment rate is 2%. After $2000 of play, the hidden pool value is at $40. If the promotion selects a winner at this point, the sum of the base amount and the hidden pool is $1040, which is put into the current pool. After summing, the hidden pool is reset to zero.

After an award is given and the current pool is established as described above, the hidden pool and the current pool are each adjusted by the delayed pool (described below), and a new lucky number is set between the value of the current pool and the maximum number.

The master server and the slave servers each maintain a delayed pool, which is not displayed to bonus promotion participants and does not contribute to the growth of the current pool during the current promotion. The slave delayed pool is comprised of play on eligible EGMs that were not able to contribute to the bonus. For example, if communication between an EGM and a slave server were lost for a period of time, restoration of communication could cause an immediate large incremental contribution from the EGM resulting from sudden realization of accrued off-line contributions. The restored EGM

therefore has a disproportionate opportunity to win the bonus because of the weighting of play involved in the selection of the winning EGM.

When an EGM goes off-line and shows an increment to the slave server after coming back on-line, the total increment goes into the slave server's delayed pool. In other words the delayed pool at each slave server is not multiplied by the contribution percentage; the entire amount of wagers that occurred off-line goes into each slave server's delayed pool. The value of the delayed pools is accrued at each slave server and periodically transmitted to the master server, which maintains the total accrued delayed pool.

The master server's delayed pool works much like the slave servers. The master server delayed pool is comprised of two components. The first component is the delayed pool contributions in dollars from disrupted EGMs. The second component is slave server contribution totals that were not able to contribute to the bonus, again perhaps because of communication difficulties between the slaves and the master. As will be recalled, these totals have already been multiplied by the contribution percentage. As a result, when communication is restored, this component cannot be added directly into the first component of the master server delayed pool. Rather, these totals must be divided by the contribution rate to convert them back to dollars.

The contribution rate, however, can vary from one bonus promotion to the next. The master server must therefore undo the calculation previously applied by the slave server prior to reporting to the master by dividing the slave's accrued contribution by the user-selected percentage from the previous bonus promotion to determine the actual dollars played while the slave server was off-line. The increment in these actual dollars that accrued before the slave went off-line is then added to the delayed pool.

Once the master server has the actual dollars played in the delayed pool—derived from the slave's delayed pools and from undoing the pool contribution calculations as just described—it is distributed to both the new current pool and hidden pool using increment rates defined for the current and hidden pools. As a result, the base amount, and the hidden and delayed pools provide the initial funding for the new bonus pool.

If applying the percentage of the delayed pool to the current pool puts the current pool over 90% of the range between the minimum and maximum numbers, the amount applied from the delayed pool to the current pool is set to 90% of this range. This defines the minimum number and creates a relatively narrow range from which the lucky number is selected. In other words, the current pool is likely to hit a winner in a relatively short time. As the games continue, the amount applied from the delayed pool to each new current pool is

limited in the same fashion until the applied percentage from the delayed pool is less than 90% of the range.

It should be appreciated that the amount of the award can be the lucky number. Or it can be a fixed amount or a noncash award, such as a car or the like.

Consideration will now be given to the features that enhance security in system 10. Generally speaking, key distribution center 29 (KDC), manages different types of keys, which will be shortly described, to encrypt digital messages on each LAN, like LAN 18A, and between each LAN and master server 14. Generally speaking, three different types of keys are used in system 10: a setup key, a master node key, and a session key. These keys are used, as will be shortly described, to authenticate and verify messages on the WAN and the LAN. Every message between master server 14 and each of the slave servers, like slave server 27, is authenticated with a signature. Likewise, all messages that authorize a bonus payment between each of the slave servers and the communications boards in their associated slot machines, like slot machine 28, 30, are also signed to authenticate the message. This will be discussed in more detail below.

Consideration will first be given to the setup key, which is a single identical key shared by each of the participating nodes, namely, each slave server, like slave server 27; master server 14; and each of the communication boards in the slot machines, like slot machines 28, 30, that may be selected to participate in the Random Rewards® bonus. The setup key can be installed by installing it in firmware associated with each node. This installation may be accomplished in several ways, including simply including it during the manufacturing or installation process in firmware memory. Preferably, one half of the setup key is encoded on each of two separate magnetic-strip cards. The setup key is then installed by requiring two individuals to swipe each card at each of the nodes, thereby downloading the setup key at each node. This procedure enhances security of the setup keys.

Next, each participating node uses the setup keys to generate a master node key that is a private key shared only between KDC 29 and the node in question. In other words, the KDC generates and stores a private key-pair between the KDC and each participating node in the entire WAN.

Shortly after each node receives the setup key, whether installed in firmware at the outset or installed after system 10 becomes operational, the node sends a request for creation of a master node key to KDC 29. This message is authenticated using a message digest encrypted with the setup key. KDC 29 responds by randomly generating a master node key that is encrypted using the setup key and a 64-bit International Data Encryption Algorithm

(IDEA™) Cyphertext Feedback (CFB). This message is also authenticated using a message digest encrypted with the setup key.

The requesting node then sends another message to the KDC, which generates a node table entry and stores the master key for that node. This message is authenticated using a message digest encrypted with the master node key. Thereafter the KDC sends a final message to the requesting node that causes it to remove all memory containing the setup key. This final message is authenticated using a message digest encrypted with the master node key.

The final key, the session key, is implemented by the KDC when it first starts up and periodically thereafter as will be shortly described. A counter, known as the session index, is associated with each session key. Each time the KDC changes session keys, it increments this counter. Whenever a node stores the session key, it associates it with the key counter and makes the key counter part of every message it encrypts. As a result, the receiving node can determine which session key was used for encryption. This is important during the time when session keys are changing, since not all nodes switch simultaneously. All nodes are periodically informed of the current session key index through a plain text message the KDC broadcasts across the entire WAN every 10 seconds. When a node detects that the broadcast session key index identifying the current key is different from the index it has stored, it requests the new session key using a session key request procedure. Once it receives the new session key, the old session key is discarded. This procedure accommodates new devices, devices that have been offline when the period for changing the session key passes (described below), and other situations in which a node does not have the current session key.

The procedure for requesting the session key starts when the requesting node sends a plain text request to KDC 29 requesting the new session key. The KDC responds with a message confirming the request authenticated using the master node key. The node responds by sending a message authenticated using the master node key. This causes the KDC to send the new session key encrypted using the master node key with a 64-bit IDEA™ CFB. It is likewise authenticated using the master node key.

In the present embodiment of the invention, the transmitting node, whether it is the master or slave server, generates the signature by computing the Secure Hash Algorithm (SHA1) over the entire contents of the message. The SHA1 results are then encrypted with a 64-bit International Data Encryption Algorithm (IDEA™) and the current session key.

It can be seen that all messages containing accumulated game play, which the master server uses to maintain the pool, and all pay commands from each slave server to its

associated slot machines, are digitally verified and authenticated. As a result, the security of system 10 is greatly enhanced.

In one aspect of the present invention, the user, via configuration workstation 15 can generate an input command to set the period for changing the session key and can modify the period by preselecting a range above and below the selected period during which the session key is randomly distributed. For example, if the user selects 600 minutes as the period for changing the session key, and the range is 10, the session key remains valid for between 590 and 610 minutes. At a randomly selected time between 590 and 610 minutes after the session key is established, a new session key is distributed. This feature further improves system security.

Also, the user, via the configuration workstation, can remove a node from the system when there is a security breach. Similarly, the user can use the configuration workstation to enter a command that overrides the automated changing of the session keys just described to require that a new session key be immediately distributed. Both of these features further enhance system security.

The present embodiment has been illustrated with a single master server having multiple slave servers connected via a WAN. More than one master server could implement bonuses on the same WAN. In other words, another master server could be added to the WAN. The additional master would have its own slave servers and maintain bonus pools that are separate from the other master or master servers.

Finally, consideration will be given to payment of other bonuses, such as promotional awards, in addition to the Random Rewards® promotion described above. The present invention can implement direct payment of these rewards across multiple properties at the slot machine including, e.g., direct transfers of redeemable credits to the slot machine; direct transfers of non-redeemable credits to the slot machine (Xtra Credit®), as described in U.S. Patent Application No. 09/134,598, filed August 14, 1998, which is hereby incorporated by reference for all purposes; and temporary change of the machine award schedule.

As in the existing single property implementations, the extension of bonusing concepts to the multi-property environment also allows more streamlined communication of promotional information. Instead of, or in addition to, more traditional communication methods such as direct mail or brochures, direct communication of promotional information can occur at the slot machine.

In accordance with some of the security enhancements described above, bonus commands, as well as other commands, may be digitally signed. These commands may flow from the master server to the slave servers or from the slave servers to the EGMs. Digital signatures allow the receiving node to verify that the commands are valid and are truly from the machine that the messages indicate.

The process of adding a digital signature typically involves the transmitting node performing a hashing function over the contents of the message, either a portion of the message or the entire contents. A hashing function, such as the SHA1 mentioned previously, converts the contents of the message into a digital 'hash.'

Generally, a hashing algorithm produces a bit string called a message digest. The message digest is a condensed representation of a message or a data file. For example, the SHA1 algorithm treats a message or a data file as a bit string. For a bit string of less than $2^{64}$ as input, SHA1 produces a 160-bit output, the message digest. The message digest is then input to a digital signature process, which then generates or verifies the signature of the message.

Using a digital signature makes it highly improbable that a message, such as a bonus command, can be altered in transit and still have a verifiable signature. In order for a message to be altered in transit and still have a message digest that produces a verifiable digital signature, the two different messages would have to produce the same message digest. This is computationally infeasible. A hacker would have to intercept the message, alter its contents, produce a new message digest, produce a digital signature matching the message digest and include all of that in the message without being discovered.

The use of digital signatures results in a message that has a message digest encrypted as a digital signature. The receiving node then uses the same algorithm on the message as was used to produce the message digest and the digital signature. The message digest and digital signatures are then compared and verified. If they match, the message is considered valid in that no other entity intercepted the message and altered its contents.

In a gaming network, such as the network discussed with reference to Figure 1, the ability to digitally sign a message provides a way to verify that the message contents have not been altered. As many of the messages in a gaming network are financial in nature with many of the parameters transmitted in various command messages relating directly to winnings, the ability to sign messages enhances the security of the system. In addition, using a digital signature may afford security to the message contents without involving levels of

cryptography that fall under export restrictions in the US. The use of digital signatures mitigates exportation concerns.

A portion of the gaming network of Figure 1 is shown in Figure 3. The portion shown is that in casino 16 and is part of LAN 18a. There is no limitation to a single property intended in the selection of this portion. The techniques and approaches discussed below may happen between a transmitting node and a receiving node at any level of the system. This includes between a master server and a slave server, either co-located at a property or with the master server located at a different site from the slave server. It also includes between a slave server and an electronic gaming machine. In addition, the electronic gaming machines 28 and 30 will more than likely be the receiving entities for most commands that are user-related, such as awarding bonuses.

As shown at slave server 26, both the transmitting and receiving nodes may have a port 262 allowing communication across the network, and a processor 264 to generate and execute commands. In addition, or as part of, the processor, the nodes will have a hashing function or module 266 and a digital signature function or module 268. It must be noted that slave server 26 may function as both a receiving node and as a transmitting node, as will be discussed further. Any transmitting node or receiving node may be configured in a like fashion to the slave server 26 show as an example here.

Several examples of the use of digital signatures in gaming networks are shown below. Possible situations in which each example may be appropriate are also given, to enhance understanding of the invention. No limitation of the applications of the invention is intended by either the examples given or the situations in which the examples may be appropriate. Any of the examples given, and any other that fall within the scope of the claims, may be used in any of the situations given, as well other situations a system designer may face.

In one example, referred to here as a direct command transmission, the master server 14 determines that a player at the machine 28 is to be awarded a bonus. The master server 14 then generates the command to pay a bonus to be sent to the EGM, as discussed above. The master server generates the bonus command and digitally signs it. The master server then transmits the message to the EGM 28. The EGM 28 then performs the hashing on the message, processes the message digest produced by the hash through a digital signature algorithm. If the signatures match, it is considered verified and the message contents can be trusted. An alternative example of the direct command transmission is where the command recipient is the slave server, not the EGM 28. This approach may be appropriate for

situations where there is a direct link between the master server and the EGMs, or the overhead of having the message received by the slave servers and processed prior to being sent to the EGMs involves too much time or computational overhead.

In another example, in a double verification process, the master server 14 transmits the message to one of the slave servers, such as 27 or 26. The slave server then verifies the signature by the above verification process. Once it has verified the signature, the slave server passes the message to the EGM 28. In this manner, an extra layer of security can be added to have the signature validated in transit between the master server and the EGM 28. This may be appropriate when the links between the master and slave servers as well as between the slave servers and the EGMs may benefit from this added layer.

A pass-through verification may offer an alternative to the double verification process, in which the slave server may merely act as a conduit for the message. If the LAN were configured a little differently, with the slave server 26 also acting as a router 20, the slave server may only pass the message to the EGM 28 without performing the signature verification. Only the EGM would perform the signature verification. The slave server merely passes the signed message through to the EGM. Situations in which there is limited hardware available to separate functions may benefit from the combination of the functions into one entity.

In another example, a slave server is the transmitting node in a two message process. This may be in response to a transmission received from the master server. If the master server sends a digitally signed message, the slave server would verify the signature of the message. The slave server would then generate the appropriate command or other message based upon the message received from the master server. The slave server would then digitally sign the new message and transmit it to the EGM, which would then verify the signature. Generation of two separate messages may decrease the likelihood of the message being altered in transit, as the message changes between the master server and the receiving machine.

An alternative to the two message process would be single, signed message process. In this process the slave server receives a communication from the master server that is not digitally signed. It may be encrypted, or in the clear. No signature verification is needed on the message from the master server. The slave server may then generate the appropriate command message, hash and sign it, and then transmit it to the EGM 28. This approach may appropriate when there is a highly secure link between the master server and the slave server, but not as good as security between the slave server and the EGM 28.

In yet another example, the slave server is the transmitting node without any precipitating communication from the master server. The slave server generates the command message, hashes and signs it and transmits it to the EGM. This may be appropriate when some measure of command is given to the slave servers, such as when the master server is located off-site.

Other situations in which the commands may be signed may include communications between the configuration server 15 shown in Figure 1 and the slave servers being assigned bonuses for rewards. Generally these servers are in secured rooms or a single room, so the digital signature may not be necessary.

A general process for use of digital signatures in a gaming network is shown in Figure 4. At 50, play begins. This will launch the bonus monitoring operation on the servers as discussed above, as well as the interaction between the player and the EGM. At 52, a triggering event occurs, such as the awarding of a bonus, a winning, etc., at 52. When the triggering event occurs, the server, such as the master server or the slave server, generates the resulting command 54. An example would be the triggering event being a payment of a bonus, and the command would be to pay out the bonus.

The command message, referred to here as a command, is then signed at 56. Again, this may be done by generating a hash of the message, creating a message digest. The message digest is then run through a digital signature algorithm, and the signature appended to the message. The command message with digital signature, referred to here as a digitally signed command, is then transmitted at 58. All of these processes from the occurrence of the triggering event to the transmission occur at the transmitting node, either the master server or the slave server.

At the EGM, the command is received at 60. The message is then hashed again, the digest and signature processes completed and the message verified. If the signature does not verify at 60, the command is rejected at 62. This may trigger some additional safeguards, such as notification of a hijacked message to a system administrator, an alarm being set off, among other security notifications.

If the signature is verified at 60, the EGM will comply with the command contained within the digitally signed message. If, for example, the command is to pay an award of $500, the award is allocated to the player, in points or money. These processes from the reception of the command to the actual execution are performed at the receiving node, such as the slave server or the EGM. The receiving node may also be referred to as a 'subservient

device,' in that slave servers are subservient to master servers and EGMs are subservient to both slave servers and master servers.

In one implementation of the invention, an existing gaming network could be upgraded with new software that would allow the master servers, slave servers and EGMs to digital sign transmissions and verify signatures on received messages. In this implementation, the embodiments of the invention may include an article of machine-readable media upon which is contained machine-readable code. The code, when executed, causes the machines, such as the master servers, slave servers and EGMs, to perform the methods of the invention.

While the invention has been shown and described with reference to a certain preferred embodiment thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the spirit and scope of the invention as defined by the appended claims.